# Tiled with LibGDX

## MAPS

- Libgdx features a generic maps API.
- All map related classes are in the com.badlogic.gdx.maps package
- This set of classes support any 2D map format.
- A map is a set of layers. A layer contains a set of objects
- Many of the supported editors allow you to specify properties on maps, layers and objects

## MAPS

### Map Layers

Layers within a map are ordered and indexed, starting by index 0. We can access the layers in some ways:

```
MapLayer layer = map.getLayers().get(0);
MapLayer layer = map.getLayers().get("my-layer");
```

These methods will return MapLayer. In our case we want a TiledMap layer so we would do it this way:

```
TiledMapTileLayer tiledLayer =
(TiledMapTileLayer)map.getLayers().get(0);
```

A layer has attributes we try to normalize, these attributes can be modified and will have an effect on how the layer is rendered:

```
String name = layer.getName();
float opacity = layer.getOpacity();
boolean isVisible = layer.isVisible();
```

## MAPS

### Map Layers

To get the objects within the layer:

```
TiledMapTileLayer tiledLayer =
(TiledMapTileLayer)map.getLayers().get(0);
```

- We will get a MapObjects, we could access by name, index or type and insert or remove them
- Tiles of a tiled map are not stored as map objects
- Objects are generally used to define trigger areas, spawn points, collision shapes and so on

## TILED MAPS

- Tiled maps are loaded into TiledMap instances
- TiledMap is a subclass of the Map class
- Maps that contain layers with tiles are handled by the classes in the com.badlogic.gdx.maps.tiled package

## TILED MAPS

### Tiled Maps Layers

Layers with tiles in them are stored in TiledMapTileLayer instances, if we want to access:

```
TiledMap tiledMap = loadMap();
TiledMapTileLayer layer =
(TiledMapTileLayer)tiledMap.getLayers().get(0); // assuming the layer at
index on contains tiles
```

- TiledMapLayer has the same attributes as MapLayer class, like properties, objects and so on.
- TiledMapTileLayer also has a two dimensional array of TiledMapTileLayer.Cell instances

The bottom left tile of a map would thus be located at (0,0), the top right tile at:

```
(tileLayer.getWidth()-1, tileLayer.getHeight()-1)
```

## TILED MAPS

### Cells

- Cells are containers for TiledMapTile
- Cells store a reference to a tile and some of its attributes
- Tiles are usually shared by multiple cells

```
Cell cell = tileLayer.getCell(column, row);
```

## TILED MAPS

### Tilesets And Tiles

- A TiledMap contain one or more TiledMapTileSet instances and this contain a number of TiledMapTile instances
- There are types kinds of tiles like static, animated or the ones created by your own implementation
- Cells within a layer can reference tiles of multiple tile sets
- It is recommended to stick to a single tile set per layer to reduce texture switches

## TILED MAPS

### Rendering Tiled Maps

- For orthogonal or top down maps, use OrthogonalTiledMapRenderer
- For isometric maps use IsometricTiledMapRenderer (is kind of experimental)
- Other renderers in this package are experimental

Creating an Orthogonal Tiled Map renderer works like this:

```
float unitScale = 1 / 16f;
OrthogonalTiledMapRenderer renderer = new
OrthogonalTiledMapRenderer(map, unitScale);
```

- The unit scale tells the renderer how many pixels map to a single world unit
- The unit scale is a way to couple your rendering coordinate system with your logical or world coordinate system

## TILED MAPS

### Loading TMX/Tiled Maps

- Tiled is a generic tile map editor that allows you to create tile layers as well as object layers
- Libgdx provides a loader to read files generated by Tiled

To load a Tiled map you have some options:

- Load it directly:

```
TiledMap map = new TmxMapLoader().load("level1.tmx");
```

This will load the file called level1.tmx from the internal file storage (the assets directory)

## TILED MAPS

### Loading TMX/Tiled Maps

- Load a file using a different file type, you have to supply a FileHandleResolver in the constructor of the TmxMapLoader:

```
TiledMap map = new TmxMapLoader(new
ExternalFileHandleResolver()).load("level1.tmx");
```

- Load a TMX map via the AssetManager, you can do the following:

```
assetManager.setLoader(TiledMap.class, new TmxMapLoader(new
InternalFileHandleResolver()));
assetManager.load("level1.tmx", TiledMap.class);

TiledMap map = assetManager.get("level1.tmx");
```

Once loaded you can treat the map just like an other TiledMap

Note: if you load your TMX map directly, you are responsible for calling TiledMap#dispose() once you no longer need it. This call will dispose of any textures loaded for the map.

## PERFORMANCE CONSIDERATIONS

While we try to make the renderers as fast as possible, there are a few things you can consider to boost rendering performance:

- Only use tiles from a single tile set in a layer. This will reduce texture binding
- Mark tiles that do not need blending as opaque. At the moment you can only do this programmatically, we will provide ways to do it in the editor or automatically
- Do not go overboard with the number of layers